##### Le langage SQL

**Table des matières**

[Introduction 3](#_Toc409596780)

[Historique 3](#_Toc409596781)

[Définition 3](#_Toc409596782)

[Le langage SQL 4](#_Toc409596783)

[Langage de Description de Données 4](#_Toc409596784)

[CREATE 4](#_Toc409596785)

[DROP 5](#_Toc409596786)

[ALTER 5](#_Toc409596787)

[Langage de Manipulation des Données 6](#_Toc409596788)

[SELECT 6](#_Toc409596789)

[INSERT 6](#_Toc409596790)

[DELETE 6](#_Toc409596791)

[UPDATE 6](#_Toc409596792)

[Sélection 8](#_Toc409596793)

[Syntaxe complète du SELECT 8](#_Toc409596794)

[Fonctions intégrées 8](#_Toc409596795)

[Opérateurs du WHERE 9](#_Toc409596796)

[Mot clé DISTINCT 9](#_Toc409596797)

[Opérateurs de partitionnement 10](#_Toc409596798)

[Jointure 11](#_Toc409596799)

[Méthode ensembliste 11](#_Toc409596800)

[Opérateurs ensemblistes 11](#_Toc409596801)

[Méthode prédicative 12](#_Toc409596802)

[Auto-jointure 14](#_Toc409596803)

[La fonction CASE 15](#_Toc409596804)

[Fonctions de manipulation des données de type date 16](#_Toc409596805)

[Fonctions de conversion 18](#_Toc409596806)

[Fonctions de traitement de chaînes 18](#_Toc409596807)

[ANNEXE : Schéma de la base VolAvion 19](#_Toc409596808)

# Introduction

Ce support a pour but de vous présenter le sous-ensemble du langage SQL utilisé pour manipuler des données au sein d’un Système de Gestion de Bases de Données Relationnel.

Les exemples donnés s’appuient sur la base VolAvion dont le schéma est donné en annexe.

## Historique

SQL est un langage structuré qui permet :

1. De créer la structure d’une base de données relationnelle (base, tables, etc.),
2. D’interroger et de modifier les données contenues dans ce type de base de données.

SQL signifie Structured Query Language. Il est issu de SEQUEL : Structured English QUEry Language, qui était le premier langage pour les S.G.B.D. Relationnels, développé par IBM en 1970 pour System R, son 1er SGBDR.

SQL a été reconnu par l’ANSI (Association de Normalisation des Systèmes d’Information) puis imposé comme norme. Il n’existe pas de S.G.B.D. Relationnel sans SQL.

Malheureusement, malgré la norme SQL, il existe un ensemble de dialectes. Les différences entre ces dialectes sont souvent minimes et tous respectent un minimum commun.

## Définition

SQL est un langage relationnel qui permet d’effectuer les tâches suivantes :

1. Définition et modification de la structure de la base de données,
2. Interrogation et modification non procédurale (c’est à dire interactive) de la base de données,
3. Contrôle de sécurité et d’intégrité de la base,
4. Sauvegarde et restauration des bases de données.

SQL est un standard qui permet d’assurer une portabilité importante des bases de données.

# Le langage SQL

## Langage de Description de Données

### CREATE

#### CREATE TABLE

Le **create table** est utilisé pour créer la description d’une table avec la liste de tous ses attributs et leur type.

CREATE TABLE PILOTE (PILNUM INT NOT NULL,

NOM VARCHAR(50),

VILLE VARCHAR(50),

UNIQUE (PILNUM));

CREATE TABLE AVION (AVNUM INT NOT NULL,

MARQUE VARCHAR(50),

TYPE VARCHAR(50),

CAPACITE INT,

LOCALISATION VARCHAR(50),

UNIQUE (AVNUM));

CREATE TABLE VOL (VOLNUM INT NOT NULL,

NUMAVION INT NOT NULL,

NUMPILOTE INT NOT NULL,

VILLEDEPART VARCHAR(50),

VILLEARRIVEE VARCHAR(50),

HEUREDEPART DECIMAL(5,2),

HEUREARRIVEE DECIMAL(5,2),

UNIQUE (VOLNUM));

#### CREATE VIEW

Le **create view** est utilisé pour :

1. obtenir une table intermédiaire constituant un extrait d’une ou plusieurs tables,
2. définir les règles de gestion permettant d’assurer le contrôle d’intégrité dans la base de données.

CREATE VIEW VOL\_NICE (VOLNUM, NUMAVION, NUMPILOTE, VILLEDEPART, VILLEARRIVEE, HEUREDEPART, HEUREARRIVEE)

AS (SELECT VOLNUM, NUMAVION, NUMPILOTE, VILLEDEPART, VILLEARRIVEE, HEUREDEPART, HEUREARRIVEE

FROM VOL

WHERE VILLEDEPART = ‘NICE’);

#### CREATE INDEX

Le **create index** sert à améliorer les performances lors de recherche dans une table sur l’attribut considéré.

CREATE INDEX VILLE\_D ON VOL (VILLEDEPART);

### DROP

Le **drop** est utilisé pour supprimer une définition de table, de vue ou d’index.

DROP TABLE VOL;

### ALTER

L’**alter** est utilisé pour modifier une définition de table, de vue ou d’index.

ALTER TABLE PILOTE ADD SALAIRE MONEY;

ALTER TABLE PILOTE ALTER COLUMN SALAIRE SMALLMONEY;

ALTER TABLE PILOTE DROP COLUMN SALAIRE;

## Langage de Manipulation des Données

### SELECT

Le **select** sert à interroger les données et à les présenter triées et/ou regroupées suivant certains critères.

#### Sélection

Il s’agit de retrouver les enregistrements qui vérifient certains critères

Exemple : trouver les vols qui arrivent après 19 heures

SELECT \*

FROM VOL

WHERE HEUREARRIVEE >= 19;

#### Projection

Il s’agit de n’afficher que certains attributs dans une table

Exemple : lister les numéros de vols

SELECT VOLNUM

FROM VOL;

### INSERT

L’**insert** permet d’ajouter un enregistrement dans une table.

INSERT

INTO AVION

VALUES (110, ‘AIRBUS’, ‘A320’, 320, ‘NICE’);

### DELETE

Le **delete** permet de supprimer un enregistrement d’une table.

DELETE

FROM AVION

WHERE CAPACITE < 100;

### UPDATE

L’**update** permet de modifier les valeurs de certains attributs d’un ou plusieurs enregistrements dans une table.

Exemple : modifier la capacité de tous les avions basés à Nice en l’accroissant de 10 %

UPDATE AVION

SET CAPACITE = CAPACITE\*1.1

WHERE LOCALISATION = ‘NICE’;

## Sélection

### Syntaxe complète du SELECT

SELECT liste des attributs à afficher

[DISTINCT] mot clé facultatif

FROM liste des noms de tables

WHERE condition sur les attributs

GROUP BY nom de l’attribut de regroupement

HAVING condition de sélection sur le regroupement

ORDER BY critère de tri

### Fonctions intégrées

Des **fonctions intégrées** peuvent être combinées à la liste des attributs.

SUM somme des valeurs de l’attribut pour les enregistrements sélectionnés

MIN minimum des valeurs de l’attribut pour les enregistrements sélectionnés

MAX maximum des valeurs de l’attribut pour les enregistrements sélectionnés

AVG moyenne des valeurs de l’attribut pour les enregistrements sélectionnés

COUNT nombre d’enregistrements sélectionnés

TOP les premiers d'une liste - associé à ORDER BY

1. Nombre d’avions dans la table

SELECT COUNT(AVNUM)

FROM AVION;

1. Capacités mini et maxi des Boeing

SELECT MIN(CAPACITE), MAX(CAPACITE)

FROM AVION

WHERE MARQUE = ‘BOEING’;

1. Capacité moyenne des avions localisés à Paris

SELECT AVG(CAPACITE)

FROM AVION

WHERE LOCALISATION = ‘PARIS’;

1. Capacité totale des avions

SELECT SUM(CAPACITE)

FROM AVION;

### Opérateurs du WHERE

Pour exprimer les conditions dans la clause **where**, on dispose de certains opérateurs :

* >, <, =, <>, <=, >= pour les comparaisons
* BETWEEN
* AND et OR
* LIKE suivi d’une expression représentant un ensemble de valeurs. Dans ces expressions % désigne un ensemble de caractères, \_ remplace une lettre
* IS NULL et IS NOT NULL : testent si un attribut possède ou non une valeur

1. Nom des pilotes dont la 2ème lettre est un S

SELECT NOM

FROM PILOTE

WHERE NOM LIKE ‘\_S%’;

### Mot clé DISTINCT

Le résultat d’un select étant un ensemble, il peut y avoir des doublons. Le mot clé **distinct** permet de préciser que l’on ne veut qu’un seul exemplaire de ces enregistrements.

1. Liste des types d’avion de plus de 250 places

SELECT TYPEAVION, CAPACITE

FROM AVION

WHERE CAPACITE > 250;

Les types A320 et B727 vont apparaître plusieurs fois.

1. Nombre d’avions en service

SELECT COUNT(DISTINCT AVION)

FROM VOL;

On compte les références distinctes.

### Opérateurs de partitionnement

#### GROUP BY

Le mot clé **group by** permet d’effectuer des regroupements sur lesquels s’opèrent les fonctions intégrées.

1. Nombre d’avions de chaque marque

SELECT MARQUE, COUNT(AVNUM)

FROM AVION

GROUP BY MARQUE;

1. Nombre de pilotes différents pour chaque avion en service

SELECT NUMAVION, COUNT(DISTINCT NUMPILOTE)

FROM VOL

GROUP BY NUMAVION;

#### HAVING

Le mot clé **having** associé au group by permet d’exprimer des conditions de sélection sur les regroupements.

1. Numéros des pilotes assurant plus d’un vol

SELECT NUMPILOTE

FROM VOL

GROUP BY NUMPILOTE

HAVING COUNT(VOLNUM) > 1;

1. Numéros des pilotes assurant plus d’un vol au départ de Nice, ainsi que le nombre de vols concernés

SELECT NUMPILOTE, COUNT(VOLNUM)

FROM VOL

WHERE VILLEDEPART = ‘NICE’

GROUP BY NUMPILOTE

HAVING COUNT(VOLNUM) > 1;

### 

## Jointure

La **jointure** consiste à rechercher entre deux tables ayant un attribut commun (même type et même domaine de définition) tous les éléments pour lesquels ces attributs ont la même valeur.

Pour représenter la jointure il y a 2 méthodes :

* la méthode ensembliste qui réalise l’intersection de deux ensembles
* la méthode prédicative qui vérifie l’égalité de deux attributs

### Méthode ensembliste

SELECT liste d’attributs

FROM table1

WHERE attribut de jointure IN (SELECT attribut de jointure

FROM table2

WHERE condition)

Le SELECT qui suit le IN est celui qui est exécuté le premier. Le résultat est un ensemble de valeurs de l’attribut de jointure. On extrait ensuite de table1 tous les enregistrements dont la valeur de cet attribut appartient à l’ensemble.

Exemple :

1. Nom des pilotes assurant un vol au départ de Paris

SELECT NOM

FROM PILOTE

WHERE PILNUM IN (SELECT NUMPILOTE

FROM VOL

WHERE VILLEDEPART = ‘PARIS’);

### Opérateurs ensemblistes

On utilise les opérateurs suivants sur les ensembles :

* IN
* NOT IN

1. Liste des avions AIRBUS allant à Paris

SELECT \*

FROM AVION

WHERE MARQUE = ‘AIRBUS’

AND AVNUM **IN** (SELECT NUMAVION

FROM VOL

WHERE VILLEARRIVEE = ‘PARIS’);

C’est à dire trouver tous les avions dont le numéro appartient à l’ensemble des numéros d’avions des vols à destination de Paris.

1. Liste des avions AIRBUS n’allant pas à Paris

SELECT \*

FROM AVION

WHERE MARQUE = ‘AIRBUS’

AND AVNUM **NOT IN** (SELECT NUMAVION

FROM VOL

WHERE VILLEARRIVEE = ‘PARIS’);

C’est à dire trouver tous les avions dont le numéro n’appartient pas à l’ensemble des numéros d’avions des vols à destination de Paris.

### Méthode prédicative

Il y a un seul SELECT pour toute la requête.

La liste de toutes les tables concernées apparaît dans le FROM.

La traduction de la jointure se fait par l’équation de jointure (égalité entre 2 attributs).

Il existe deux méthodes pour exprimer les conditions de jointure.

Une première méthode qui consiste à exprimer les conditions de jointure dans la clause WHERE. Elle existe encore sur certains systèmes mais doit être bannie sur SQL Server 7 et ses successeurs, tout comme sur Access, car elle est non conforme aux nouvelles normes ANSI 92.

Une deuxième méthode, à utiliser dans tous les cas, car elle est implémentée notamment sur SQL Server 7 et Access, consiste à exprimer les jointures dans la clause FROM.

1. N°, type et capacité des avions en service

Ancienne méthode :

SELECT AVNUM, MARQUE, CAPACITE

FROM AVION, VOL

WHERE AVNUM = NUMAVION

Nouvelle méthode

SELECT AVNUM, MARQUE, CAPACITE

FROM AVION INNER JOIN VOL ON AVNUM = NUMAVION

La clause INNER JOIN, jointure interne, ne retient que les lignes des deux tables pour lesquelles l'expression exprimée au niveau de ON se vérifie.

Il existe aussi des jointures externes qui permettent de retenir, en plus des lignes correspondantes à l'expression exprimée en ON, les lignes qui dans la table de gauche ou dans la table de droite, n'ont pas de correspondance. Elles se nomment jointures externes (OUTER) droite (RIGHT) ou gauche (LEFT)

1. Si nous voulons la liste de tous les avions avec, pour ceux qui volent, des informations sur les villes de départ et d'arrivée, nous écrierons :

SELECT AVNUM, MARQUE, CAPACITE, VILLEDEPART, VILLEARRIVEE

FROM AVION LEFT OUTER JOIN VOL ON AVNUM = NUMAVION

Pour les avions qui ne sont pas en service, les attributs VILLEDEPART et VILLEARRIVEE prendront la valeur NULL

Compléments

INNER

Spécifie toutes les paires correspondantes de lignes renvoyées. Supprime les lignes n'ayant pas de correspondance entre les deux tables. Ceci est l'option par défaut si aucun type de jointure n'est spécifié.

FULL [OUTER]

Précise qu'une ligne de la table de gauche ou de droite, qui ne correspond pas à la condition de jointures, est comprise dans l'ensemble de résultats et que les colonnes de sortie qui correspondent à l'autre table ont des valeurs NULL. Ceci est fourni en plus de toutes les lignes généralement renvoyées par la fonction INNER JOIN.

LEFT [OUTER]

Spécifie que toutes les lignes de la table de gauche ne respectant pas la condition de jointure sont comprises dans l'ensemble de résultats, et que les colonnes de sortie de l'autre table ont des valeurs NULL en plus de toutes les lignes renvoyées par la jointure interne.

RIGHT [OUTER]

Spécifie que toutes les lignes de la table de droite ne respectant pas la condition de jointure sont comprises dans l'ensemble de résultats, et que les colonnes de sortie correspondant à l'autre table ont des valeurs NULL en plus de toutes les lignes renvoyées par la jointure interne.

JOIN

Indique que l'opération de jointure spécifiée doit avoir lieu entre les tables ou vues données.

ON <condition de recherche>

Indique la condition sur laquelle se base la jointure. La condition peut préciser un prédicat quelconque, bien que des colonnes et des opérateurs de comparaison soient souvent utilisés, par exemple :

Lorsqu'une condition spécifie des colonnes, elles ne doivent pas nécessairement avoir le même nom ou le même type de données ; cependant, si les types de données sont différents, ils doivent être compatibles ou pouvoir être convertis de manière implicite par Microsoft SQL Server. Si les types de données ne peuvent pas être convertis implicitement, la condition doit le faire explicitement à l'aide de la fonction CAST.

### Auto-jointure

L’**auto-jointure** est la jointure entre une table et elle-même, pour sélectionner des enregistrements correspondant à d’autres de la même table. Il est nécessaire de recourir alors à des alias pour définir la table déjà utilisée.

Exemple :

1. Type des avions ayant la même capacité

SELECT AVION.TYPEAVION, AVION2.TYPEAVION, AVION.CAPACITE

FROM AVION INNER JOIN AVION AS AVION2

ON AVION.CAPACITE = AVION2.CAPACITE

WHERE AVION.AVNUM <> AVION2.AVNUM

## La fonction CASE

Elle permet l’évaluation successive de différentes conditions au sein d’un même groupe.

La fonction CASE peut être mise en œuvre de deux manières :

* La fonction CASE détermine le résultat en comparant une expression à un jeu d'expressions simples ;
* La fonction CASE détermine le résultat en évaluant un jeu d'expressions booléennes.

Les deux formes prennent en charge un argument ELSE facultatif.

Les deux exemples qui suivent vous permettent de mieux appréhender la syntaxe de ces expressions conditionnelles.

A noter : lorsque vous recourez à cette structure pour réaliser des mises à jour conditionnelles, il faut prendre garde au fait que la colonne devant être modifiée vaudra Null si vous ne lui affectez aucune valeur et ne conservera donc pas sa valeur initiale.

D’où la programmation de l’instruction sur ELSE qui sera exécutée dans le cas où aucune condition n’a été vérifiée dans l’expression des différents cas.

Exemples :

* Mettre à jour les salaires des pilotes n° 1 et 2 respectivement à 12 500 € et 25 000 € , les autres salaires restent inchangés

UPDATE PILOTE

SET SALAIREBRUT =

CASE PILNUM

WHEN 2 THEN 12500

WHEN 3 THEN 25000

ELSE SALAIREBRUT

END

* Augmenter de 10 % les salaires des pilotes gagnant moins de 10 000 €, de 8 % ceux compris entre 10 000 et 20 000 € et de 5 % les autres

UPDATE PILOTE

SET SALAIREBRUT =

CASE

WHEN SALAIREBRUT < 10000 THEN SALAIREBRUT \* 1.1

WHEN SALAIREBRUT BETWEEN 10000 AND 20000 THEN SALAIREBRUT \* 1.08

ELSE SALAIREBRUT \* 1.05

END

## Fonctions de manipulation des données de type date

Elles permettent d'effectuer des opérations sur les données de type Date et Datetime.

|  |  |
| --- | --- |
| DATEADD | Ajout d’un intervalle de temps à une date |
| DATEDIFF | Intervalle de temps entre deux dates |
| DATEPART | Extraction d’une partie de date |
| DATENAME | Chaîne représentant une partie de date |
| DAY, MONTH, YEAR | Renvoi d’une partie de date |
| GETDATE, GETUTCDATE | Date du système |

#### DATEDIFF(partie\_de\_date,date\_début,date\_fin)

Elle renvoie la différence entre deux dates. Le type des valeurs du résultat est : Integer.

*partie\_de\_date*

Paramètre qui indique l'élément de date sur lequel calculer la différence. Le tableau suivant répertorie les éléments de date et les abréviations que reconnaît Microsoft® SQL Server™.

Partie de date Abréviations

year aa, aaaa

quart qq, q

month mm, m

dayofyear ja, a

day jj, j

week wk, ww

hour hh

minute mi, n

second ss, s

millisecond ms

*date\_début*

Date de départ pour le calcul.

**date\_début** est une expression correspondant à une valeur de type datetime ou smalldatetime, ou encore à une chaîne de caractères dans un format de date.

Étant donné que la minute est la précision de smalldatetime, en cas d'utilisation d'une valeur smalldatetime, les secondes et les millisecondes sont toujours égales à 0.

Si vous précisez seulement les deux derniers chiffres de l'année, les valeurs inférieures ou égales aux deux derniers chiffres de la valeur de l’option de configuration coupure des années à deux chiffres sont considérées dans le même siècle que l'année de coupure. Les valeurs supérieures aux deux derniers chiffres de la valeur de cette option sont considérées dans le siècle qui précède l'année de coupure. Par exemple, si l'option coupure des années à deux chiffres a pour valeur 2049 (valeur par défaut), 49 est interprété comme 2049 et 2050 est interprété comme 1950. Pour éviter toute ambiguïté, rédigez les années à l'aide de quatre chiffres.

Pour plus d'informations sur la spécification des valeurs de temps, voir Formats d'heure. Pour plus d'informations sur la spécification des dates, voir datetime et smalldatetime.

*date\_fin*

Date de fin pour le calcul.

**date\_fin** est une expression correspondant à une valeur de type datetime ou smalldatetime, ou encore à une chaîne de caractères dans un format de date.

**Nota :**

date\_début est soustrait de date\_fin. Si date\_début est une date plus tardive que date\_fin, une valeur négative est renvoyée.

DATEDIFF renvoie une erreur si le résultat dépasse les limites valides pour les valeurs entières. Pour les millisecondes, le nombre maximal qui peut être représenté correspond à 24 jours, 20 heures, 31 minutes et 23,647 secondes. Pour les secondes, le nombre maximal qui peut être représenté correspond approximativement à 68 années.

La méthode utilisée pour compter les limites traversées comme les minutes, les secondes et les millisecondes assure la cohérence du résultat donné par DATEDIFF pour tous les types de données. Le résultat est une valeur entière signée, égale au nombre de limites d'éléments de date traversés entre la première et la seconde date. Par exemple, le nombre de semaines entre le dimanche 4 janvier et le dimanche 11 janvier est égal à 1.

Exemples :

Nombre de jours entre la date de naissance et la date du jour.

SELECT NOM, DATEDIFF(DAY,DATENAISSANCE,GETDATE()) as "Nombre de jours depuis la naissance"

FROM PILOTE

#### DATEADD(partie\_de\_date,date\_début,date\_fin)

Cette fonction s’utilise de la même façon que la fonction DATEDIFF.

Exemples :

Ajout de 3 jours à la date de naissance du Pilote

SELECT DATEADD(DAY,3, DATENAISSANCE) as "Date + 3 jours", DATENAISSANCE

FROM PILOTE

#### DATEPART et DATENAME

Ces fonctions permettent d’extraire des portions d’une date

Exemples :

Extrait de portions de la date de naissance

SELECT DATEPART(DAYOFYEAR,DATENAISSANCE), DATEPART(DW,DATENAISANCE), DATEPART(MONTH,DATENAISSANCE), DATEPART(DW,DATENAISANCE), DATENAME(MONTH,DATENAISSANCE)

FROM PILOTE

## Fonctions de conversion

Certaines conversions ne peuvent être automatiquement réalisées par le système. Nous devons alors réaliser ces conversions de manière explicite au moyen de la fonction de conversion CONVERT.

Attention aux types d’origine et résultant de la conversion : toutes les combinaisons ne sont pas admises.

CONVERT permet de définir un style pour la donnée convertie.

La fonction système GETDATE() renvoie la date du jour.

Si je souhaite convertir celle-ci dans un format américain, j’utilise la fonction CONVERT avec le style approprié.

Pour plus d’informations, voir l’aide de Transact SQL à l’index CONVERT.

SELECT CONVERT(DATETIME,GETDATE(),102) AS "Date au format américain"

Je souhaite que le Salaire brut soit converti et présenté dans un format décimal de 10 de long avec 3 chiffres après la virgule.

SELECT NOM, SALAIREBRUT, CONVERT(DECIMAL(10,3),SALAIREBRUT))

FROM PILOTE

## Fonctions de traitement de chaînes

Quelques exemples dans ce tableau, car elles sont nombreuses…

Voir aide à l’index fonctions, chaîne

|  |  |
| --- | --- |
| LEFT, RIGHT | Extraire des caractères à gauche ou à droite |
| UPPER, LOWER | Mettre en majuscules ou minuscules |
| LTRIM, RTRIM | Suppression des espaces à gauche ou à droite |
| SUBSTRING | Extraction d’une sous chaîne |
| REVERSE | Inversion d’une chaîne (miroir…) |
| LEN | Longueur d’une chaîne |
| ASCII | Valeur ascii d’un caractère |
| NCHAR | Renvoie le caractère Unicode fonction de la valeur donnée |
| REPLACE | Remplacement d’une occurrence de chaîne par une autre |

Liste des noms des pilotes formatés.

Le premier caractère de gauche est mis en majuscules

Les autres caractères en minuscules

SELECT UPPER(SUBSTRING(NOM,1,1)) + SUBSTRING(NOM,2,Len(NOM)-1)

FROM PILOTE

Remplacement de l’occurrence Toulouse par Ville Rose dans l’attribut Ville de la Table Pilote.

SELECT REPLACE(VILLE,'Toulouse','Ville Rose')

FROM PILOTE

WHERE VILLE LIKE 'TOULO%'

## ANNEXE : Schéma de la base VolAvion

